1. 这个论文没有提到本车运动转换的问题，应该是没有进行处理的
2. 这个论文没有提到对目标是否预测了速度的问题，从评估结果来看没有，因此是一个多帧叠加但仅预测位置的网络

DynStatF: An Efficient Feature Fusion Strategy for LiDAR 3D Object Detection

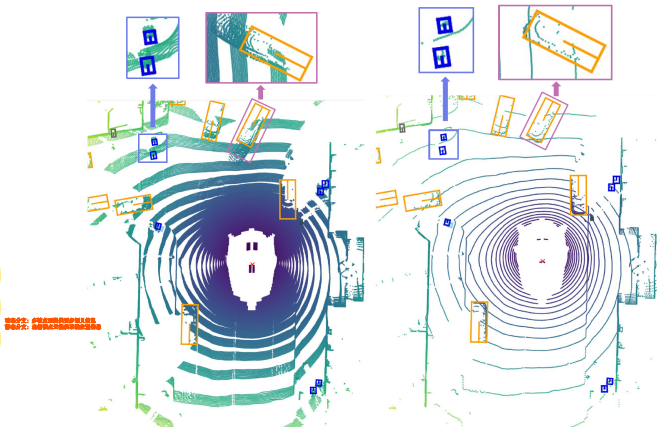
Yao Rong[1*], Xiangyu Wei[2*], Tianwei Lin[2], Yueyu Wang[2], Enkelejda Kasneci[3]
[1]University of Tübingen, [2]Horizon Robotics, [3]University of Munich

Abstract

*Augmenting LiDAR input with multiple previous frames provides richer semantic information and thus boosts performance in 3D object detection, However, crowded point clouds in multi-frames can hurt the precise position information due to the motion blur and inaccurate point projection. In this work, we propose a novel feature fusion strategy, DynStaF (**Dyn**amic-**Sta**tic **F**usion), which enhances the rich semantic information provided by the multi-frame (dynamic branch) with the accurate location information from the current single-frame (static branch). To effectively extract and aggregate complimentary features, DynStaF contains two modules, Neighborhood Cross Attention (NCA) and Dynamic-Static Interaction (DSI), operating through a dual pathway architecture. NCA takes the features in the static branch as queries and the features in the dynamic branch as keys (values). When computing the attention, we address the sparsity of point clouds and take only neighborhood positions into consideration. NCA fuses two features at different feature map scales, followed by DSI providing the comprehensive interaction. To analyze our proposed strategy DynStaF, we conduct extensive experiments on the nuScenes dataset. On the test set, DynStaF increases the performance of PointPillars in NDS by a large margin from 57.7% to 61.6%. When combined with CenterPoint, our framework achieves 61.0% mAP and 67.7% NDS, leading to state-of-the-art performance without bells and whistles.*

1. Introduction

LiDAR sensor is widely used for 3D object detection in the context of autonomous driving because of its high precision in depth information. Recent methods based on LiDAR information utilizing Bird's Eye View (BEV) can be mainly categorized into two groups: voxel-based (VoxelNet proposed by Zhou and Tuzel [27]) and pillar-based (PointPillar proposed by Lang et al. [11]). The former group [7, 23, 24, 27] first divides points in the space into equally distributed voxels, and obtain features with several 3D convectional layers. The latter one [6, 11, 20, 21] converts 3D points into pseudo images by generating pillars at each position in a 2D image, whose size on the vertical-axis being equal to the whole available space and thus is able to directly acquire feature representations using 2D convolution. Without feature compression on the vertical-axis, voxel-based methods yield higher performance, while pillar-based models are more efficient in the computation and preferred in real-time applications.



(a) Multi-frame input. (b) Single-frame input.

Figure 1. Visualization of multi-frame (10 sweeps) and single-frame input. Bounding boxes are ground-truth objects on nuScenes. Zoom-in images above demonstrate a clear view.

LiDAR point cloud contains precise geometric shapes and exact positions of objects, but it suffers from the irregular point density: points are dense in the area closed to the LiDAR sensor while very sparse far away. Detecting objects with fewer points is very difficult. Suggested by [2, 12], using multiple LiDAR sweeps (frames) provides richer point cloud information to eliminate the unclarity due to the sparsity of the points. Points from multiple sweeps are aggregated directly and distinguished by expanding input data with relative timestamp information as an extra dimension, which also enhances the network with valuable temporal information. Figure 1 illustrates the difference between the multi-frame and single-frame input. In this scene, there are several vehicles and pedestrians in front of the car. It is easy to determine the location of objects based on the

*These authors contributed equally to this work

unambiguous points on their surface with a single sweep. However, after accumulating ten sweeps of point clouds, motion blur is observed around vehicles and pedestrians such that edges of moving objects become obscure to be accurately recognized (see zoom-in images), leading to confusion about concrete positions. In a word, multi-frame LiDAR input boosts recognition performance by augmenting the input with meaningful motion characteristics (*dynamic* information), but suppresses the advantage of a single frame in exact object localization capability (*static* information).

However, existing works commonly employs only multi-frame point cloud data as input, such as [5, 7, 10, 29] conducting experiments on the large-scale outdoor dataset nuScenes [2]. Based on the observations above, we propose a novel unified framework named DynStaF, standing for Dynamic-Static Fusion, to bridge the current research gap by fusing the rich semantic information provided by the multi-frame input with the accurate location information from the single-frame data effectively. To the best of our knowledge, DynStaF is the first attempt to deploy a two-stream architecture for extracting and fusing features from multi-frame and single-frame LiDAR input.

DynStaF deploys a dual pathway architecture to operate on BEV features from both input types concurrently across the 2D backbone. To address the feature interaction, we introduce two fusion modules, Neighborhood Cross Attention (NCA) and Dynamic-Static Interaction (DSI) performing feature fusion between two branches at different levels. An attention mechanism is adapted to produce the cross attention, but a vanilla cross attention module computes the attention matrix globally. LiDAR BEV feature maps are sparse where correlative features are distributed locally. Considering this characteristic of BEV features, we do not need to compute global attention, as it does not bring significant benefits but introduce an overhead of computation. Thus, we choose to conduct cross-attention limited to the neighborhood area. Concretely, NCA regards features from the single-frame branch as queries and obtains keys and values in the neighborhood of queries from the multi-frame feature map. After several blocks in the backbone, the feature maps become dense. At this stage, we utilize the CNN-based DSI module which conducts comprehensive interaction at each pixel position. The fused feature contains rich semantic context and accurate position information that promotes the detection precision.

In summary, our work has three main contributions:

- We propose a novel feature fusion strategy termed as DynStaF which has a dual pathway architecture to efficiently fuse the complementary information from the multi-frame and single-frame LiDAR input.

- Taking into account the specific features of BEV feature maps extracted from dynamic and static branches, we introduce two modules designed for fusion at distinct levels. Neighborhood Cross Attention module is designed for sparse feature maps, while Dynamic-Static Interaction module for dense feature maps.

- We conduct extensive experiments to analyze and benchmark our methods on the challenging dataset nuScenes. DynStaF boosts the performance of PointPillars [11] significantly on the nuScenes test set by 3.9% in NDS and 5.9% in mAP. When using CenterPoint [26] as the backbone, our framework achieves 67.7% and 61.0% in NDS and mAP, respectively, surpassing other state-of-the-art methods without bells and whistles.

2. Related Work

3D object detection with LiDAR. The task of 3D object detection based on LiDAR in autonomous driving recently is to detect traffic participants and place 3D bounding boxes around them. Along with the detection, classes as well as attributes of objects (e.g., moving or parked) or other information are estimated [2, 18]. Recent algorithms for LiDAR 3D object detection are all based on BEV feature maps. VoxelNet [27] turns point clouds into voxels and apply first 3D CNNs to encode the voxel features and then 2D convolutions to accomplish the detection. Besides, SECOND is another popular framework deploying 3D convolution operations [23]. Lang et al. [11] propose to encode the point clouds to pillar vectors and then project these pillars onto the 2D BEV space. Frameworks based on PointPillars only need 2D convolutional layers to process the point BEV feature maps for 3D object detection [6, 11, 20, 21]. To further boost the performance of detectors, CenterPoint [26] proposes a new detection head, which first detects object centers of and then computes other attributes such as bounding box sizes, followed by refining these estimates in the second phase. It turns to be effective when combining with a 3D backbone such as VoxelNet, which is widely used as a state-of-the-art framework. In this work, we use PointPillars and CenterPoint as our 2D and 3D backbones to show the effectiveness and the compatibility of our DynStaF.

Feature fusion strategy. Feature fusion can boost the performance in 3D LiDAR object detection. Multi-modality fusion is one popular strategy, for example, [1, 12, 14] propose frameworks that fuse camera and LiDAR data, where the performance is stronger than using a single modality. Another group of feature fusion works does not require multiple sensors, for instance, Deng et al. [7] fuse BEV features with RV (range view) features, as RV provides dense features while BEV features are sparse but not overlapped. Combining two views improves the performance as it gives comprehensive spatial context. HVPR

(Hybrid Voxel-Point Representation) [15] utilizes voxel-based features and point-based features as used in PointNet++ [17]. In this way, voxel features, which are effective to be extracted, are integrated with more accurate 3D structures from point streams. As geometric information gets lost when projecting to the 2D BEV space, MDRNet [10] enriches the BEV features with voxel features to keep the geometry information. In our project, we propose a novel feature fusion strategy based on the characteristics of multi-frame and single-frame LiDAR input, which keeps features in both branches interacting across the whole BEV feature processing. Our strategy is trained end-to-end on the 2D BEV space, which can be directly applied to any state-of-the-art architectures to boost the performance.

Transformer attention mechanism. Thanks to the attention mechanism, the transformer architecture is powerful in fusing features from different source or modalities for 3D object detection or other tasks in autonomous driving [1, 7, 12, 13, 28]. In [7], the authors use the BEV features as queries and RV features as keys and values to conduct the cross attention between the two views. TransFusion [1] designs a new detection head based on transformer. In the first stage, a sparse set of object queries from LiDAR BEV features are used to get the initial bounding boxes; In the second stage, another transformer layer is deployed to obtain the cross attention from camera images and LiDAR data. Centerformer [28] enhances the center-based object detection by using the center candidates as queries in a DETR-style (DEtection TRansformer [3]) transformer. Moreover, cross attention between current frame and previous frames are extracted using a deformable DETR [30]. Li et al. [13] also deploy deformable DETR to gain the temporal and spatial attention among multi-camera images for the object detection. Different from previous work, our method adopts the neighborhood attention mechanism for transformers [8] to get the cross attention between multi-frame and single-frame LiDAR input. As the BEV features are sparse, the object should be at a similar spatial location in both input and thus focusing on neighborhood produces high-quality fusion, which is verified by our experiment results.

3. Method

Most recent LiDAR-based 3D detection approaches aggregate raw point clouds from a sequence of LiDAR point clouds and use the (relative) timestamp as an additional feature dimension to improve detection performance. This setting is effective in compensating the sparsity of point clouds with a single frame as input for 3D object detection. As discussed in Section 1, point clouds from previous frames will bring ambiguity in localization especially for moving objects in crowded scenarios. To mitigate this adverse impact, we propose to deploy cross attention to efficiently fuse spatio-temporal semantic features from input sequence with the accurate localization information from the current frame. A dual pathway architecture is designed to process current and aggregated point clouds separately, where the extracted features are fused progressively. Our framework is termed as "DynStaF", and we refer the multi-frame branch as "Dynamic Branch" and the single-frame branch as "Static Branch" to highlight the rich motion information and accurate location information in each branch.

3.1. Overall Architecture

We follow general 3D LiDAR object detector settings without requiring any extra input information. Popular detectors such as pillar-based frameworks project point cloud into BEV feature space after voxelization (Voxel Feature Encoding), while voxel-based frameworks usually process the voxels with a 3D backbone additionally. All popular pillar-based/voxel-based architectures can be straightly deployed as the dynamic branch in DynStaF. Complicated 3D backbones may be used to process voxels to obtain BEV features such as in CenterPoint [26]. However, the static branch is designed to be light-weighted and only needs VFE to encode the BEV features, i.e., no 3D backbone is needed in the static branch. DynStaF operates on BEV features. The projected BEV feature map of dynamic branch is denoted as $F_d \in \mathbb{R}^{C_d \times W_d \times H_d}$ and $F_s \in \mathbb{R}^{C_s \times W_s \times H_s}$ for static branch, where C, W and H refer to the number of channel, width and height of the generated BEV maps, respectively. Before starting the feature fusion, F_s is processed with extra convolutional layers to reach the same dimension as F_d if their dimensions vary. Feature fusion between two branches occurs regressively using NCA in our DynStaF, as illustrate in Figure 2 (Upper).

In the l-th fusion block ($l \in \{1, 2, ..., N\}$), given the input dynamic branch feature F_d^l and static branch feature F_s^l, the output can be formulated as:

$$F_s^{l+1} = \mathcal{B}^l(\mathcal{A}^l(F_d^l, F_s^l)) \tag{1}$$

where $\mathcal{A}^l(\cdot)$ is the NCA module and $\mathcal{B}^l(\cdot)$ refers to the CNN block. In the dynamic branch, F_d^l is processed only by the 2D CNN blocks as it is in the original backbone. After these two operations, the output F_s^l is designed to be in the same dimension as F_d^l for each layer. After all N blocks (N is identical to the number of blocks in the dynamic 2D backbone), the feature maps F_d^{out} and F_s^{out} have smaller sizes, i.e., the features are dense compared to the BEV features in the beginning. At this stage, the DSI module enhances the interaction between two features, whose output is fed to the rest of the pipeline for object detection.

3.2. Dynamic-Static Fusion Module

Neighborhood Cross Attention (NCA) module. Considering that BEV features from the static branch provides

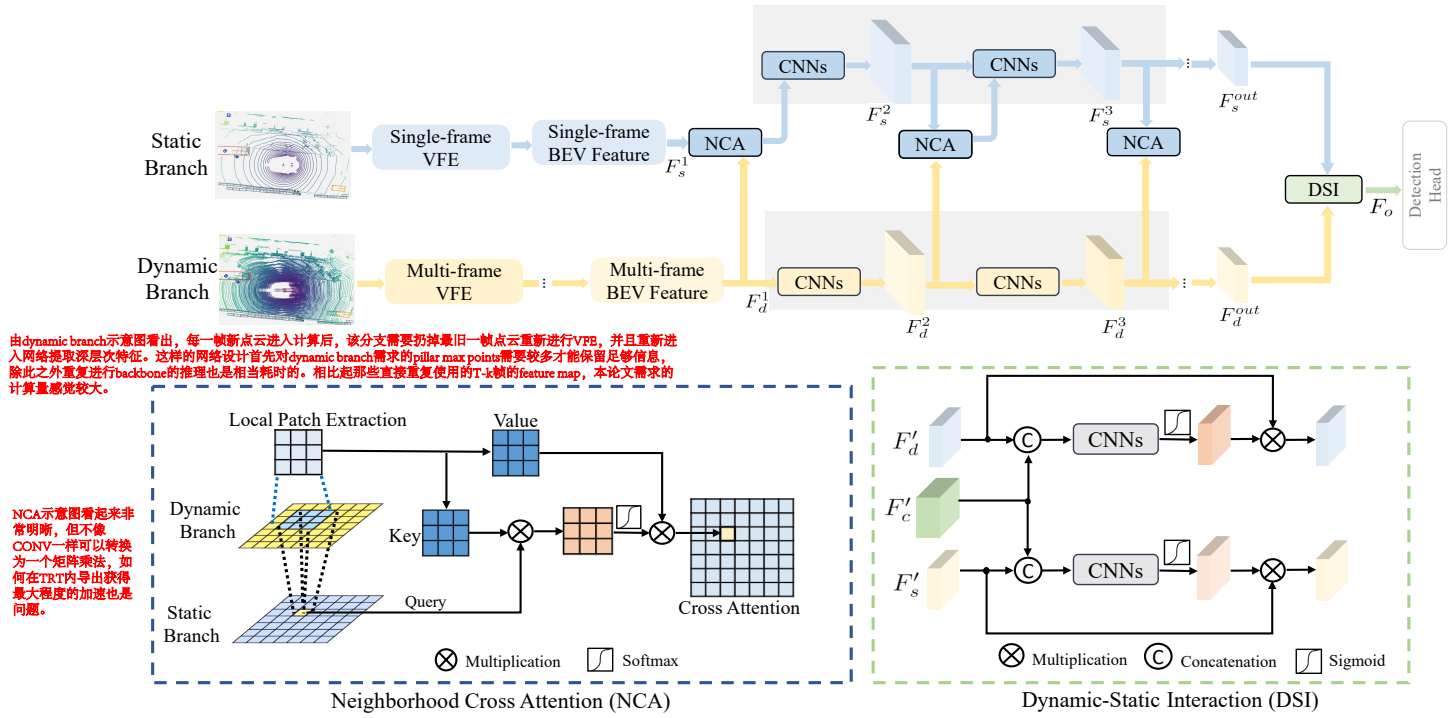


Figure 2. **Upper**: Overall Architecture of DynStaF. 2D convolutional backbone is highlighted with the gray color. **Bottom**: The overview of two core fusion modules: Neighborhood Cross Attention (NCA) and Dynamic-Static Interaction (DSI). Channel dimension in NCA is omitted for a clear view.

precise object location information and the rich spatio-temporal semantic information can be found in dynamic branch, we use features in F_s^l as queries and generate keys and values from F_d^l to achieve the cross attention. As relevant features for the same object should locate at a similar position in both features, we argue that the local information in the neighborhood of a specific query is more essential to build the cross attention in the context of BEV feature maps. Moreover, BEV feature maps are relatively large but sparse, where only a few number of pixels are occupied with non-empty pillars. Limiting a neighborhood helps save computational cost as well. We adapt Neighborhood Attention Transformer proposed for establishing self attention in the image classification task in [8] to our purpose of building cross attention. The illustration of the cross attention is shown in Figure 2 (Bottom Left).

Concretely, we first tokenize the BEV feature map using convolutional layers and denote it as a sequence of m-dim feature vectors, for instance the feature sequence from the static branch is $F_s \in \mathbb{R}^{n \times m}$. The tokenized feature F_s is linearly projected to a query $Q_s \in \mathbb{R}^{n \times q}$. For the tokenized feature sequence in the multi-frame branch, it is projected to the key $K_d \in \mathbb{R}^{n \times q}$ and the value $V_d \in \mathbb{R}^{n \times v}$ using a linear layer. The cross attention A_c for a query i in the single-frame feature map is calculated as:

$$A_c^i = \sigma\left(\frac{Q_s^i \cdot (K_d^{\rho(i)})^T + B^{(i,\rho(i))}}{\sqrt{v}}\right) \cdot V_d^i \qquad (2)$$

where $\rho(i)$ is the neighborhood with the size of k centered at the same position in the multi-frame branch, $B^{(i,\rho(i))}$ denotes the positional bias added to the attention and σ refers to SoftMax. When multi-headed attention is applied, the outputs of each head are concatenated. For each pixel in the feature map, we calculate the cross attention as above.

Another linear layer is added on top of the A_c^i. A shortcut and two extra linear layers are utilized to further process this output. To enhance the features with accurate position information, we compute the self attention of the single-frame branch using the same algorithm, but use the linear projection to obtain queries, keys and values all from the single-frame feature. The concatenation of the outputs from the cross attention and self attention is fed into a convolutional layer, leading to the final output of the NCA module. The complete operation of NCA is depicted in Figure 3.

Dynamic-Static Interaction (DSI) module. After processed by the CNN and NCA blocks, feature maps become dense and rich in information. Although the static branch is already enhanced with local features from dynamic branch

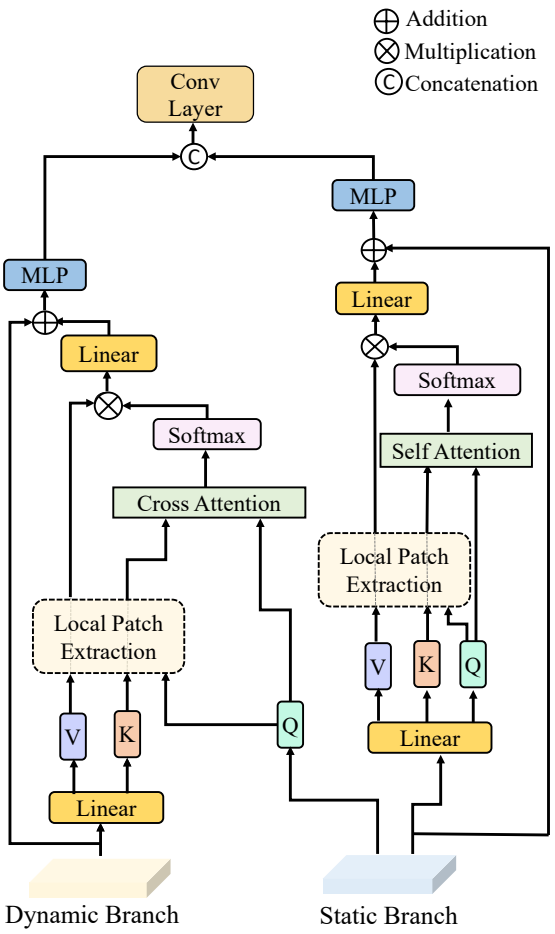


Figure 3. Illustration of Neighborhood Cross Attention (NCA) module. Input is feature maps from two branches F_s^l and F_d^l.

by NCA, it cannot guarantee to keep all detailed semantic knowledge of the objects. Therefore, we add an interaction module before the detection head to sufficiently consolidate features from two branches. Given the features from the single-frame branch denoted as $F_s^{out} \in \mathbb{R}^{C\times W\times H}$ and $F_d^{out} \in \mathbb{R}^{C\times W\times H}$ from the multi-frame branch, the concatenation of both feature maps $F_c \in \mathbb{R}^{2C\times W\times H}$ is used to guide the interaction as it contains a comprehensive view of both features. Specifically, three convolutional layers first process F_c, F_s^{out} and F_d^{out} separately, whose output here denote as F_c', F_s' and F_d'. DSI takes them as input and produces two feature maps for each branch using CNN blocks as depicted in Figure 2 (Bottom Right). The two output components of DSI is then concatenated together with F_c', followed by another CNN block to produce the output $F_o \in \mathbb{R}^{C\times W\times H}$, which is fed into the detection head.

4. Experiments

4.1. Implementation details

Dataset. We conduct our experiments on nuScenes [2], which is a large-scale dataset collected in the real-world containing multiple sensor data. In this work, we only use the LiDAR data (with the frequency of 20 FPS) to tackle with the 3D object detection task. In total, there are 700 video sequences in the training set, 150 videos in validation and test sets each. Following most of the previous works [1, 2, 7], we use 10 sweeps for the multi-frame input, corresponding to the LiDAR information in the previous 0.5s. Ten object categories ranging from cars, pedestrians to traffic cones are labelled.

Architecture details. We test our DynStaF strategy on two popular frameworks, Pointpillars [11] and CenterPoint [26]. When adding DynStaF to the PointPillars, we use three NCA modules to get the fused features, the same number of CNN blocks in the 2D backbone as in the original framework. CenterPoint network has two 2D convolutional blocks before the CenterPoint Head, where DynStaF is plugged in. As CenterPoint contains 3D backbone to process voxels, we reduce the channel number in the 2D backbone by half to keep the comparable computation cost. For each NCA module, given the corresponding CNN block in the dynamic branch with the input feature size of $c_i \times w_i \times h_i$ and the output size of $c_o \times w_o \times h_o$, we first use two convolutional layers to tokenize the features (i.e., F_d^i and F_s^i) into the features with the size $\frac{c_o}{2} \times w_i \times h_i$. Then, the cross attention is calculated with the neighborhood range set to 7 and the number of attention heads to 8. Each NCA module produces features with the same output size of $c_o \times w_o \times h_o$.

Training Loss We use the anchor-based loss proposed in [11] for training the PointPillars-based model. The loss is the weighted sum of three components: localization loss (L1 loss), classification loss (focal loss) and direction loss (cross-entropy loss), whose weights are 0.25, 1.0, 0.2, respectively. The loss used to train the CenterPoint-based model is anchor-free [26], which contains classification loss and regression loss. The former one is the cross-entropy loss between the predicted and ground-truth labels with the weight of 1, and the latter one is the L1 regression loss of bounding boxes with the weight of 0.25.

Training details. For PointPillars-based models, the point range is set to [-51.2m, 51.2m] for x-/y-axis and [-5, 3] for z-axis. During training, points are randomly flipped along x- and y-axis. Random rotation with a range of [-$\frac{\pi}{8}$, $\frac{\pi}{8}$] around the z-axis is applied. Moreover, a random global scaling factor is set in the range of [0.95, 1.05]. When uisng CenterPoint with the voxel size of (0.075m, 0.075m,

0.2m), random rotation along z-axis is set to [-$\frac{\pi}{4}$, $\frac{\pi}{4}$]. Class-balanced sampling [29] is deployed in all training. We follow the same training scheme used in [11, 19, 26]. Our experiments are conducted under the framework OpenPCDet [19] and all models are trained for 20 epochs with the batch size of 32 on 8 V100 GPUs.

Evaluation metrics. Following the nuScenes benchmark for the detection task [2], evaluation metrics used in our experiments include mAP (mean Average Precision) and a set of True Positive metrics (TP metrics). When calculating mAP, the criterion for matching between prediction and ground-truth is the 2D center distance on the ground plane. The final mAP score is averaged over all thresholds and all classes. A match in TP metrics is defined as the center distance is inside 2m. There are five TP metrics and the final score for each metric is averaged over all classes (ATE, ASE, AOE, AVE, and AAE measuring translation, scale, orientation, velocity, and attribute errors, respectively). As different metrics capture different performance aspects, a nuScenes detection score (NDS) is defined by combining all these metrics together.

4.2. Comparison with other methods

Results on nuScenes validation set. We train our model on the nuScenes training set and evaluate on the validation set. Table 2 reports the comparison with other state-of-the-art methods. We use mAP and NDS as evaluation metrics. To fairly benchmark different results, we also consider the performance gain of each strategy compared to its own backbone model reported in the original paper, as models may vary in performance with differently trained backbone models. Compared to our re-implemented PointPillars using the multi-frame as input, our DynStaF improves mAP by 5.8% and NDS by 4.1%. [1, 7, 28] deploy CenterPoint [26] as their backbone model. We see that our DynStaF achieves the most performance gain in both metrics compared to all other SOTA methods using the CenterPoint as the backbone. Our CP+DynStaF reaches 67.1% and 58.9% on NDS and mAP, respectively, which achieves the best performance on the validation set. Performance gain is significant on both backbone models, highlighting the compatibility of DynStaF.

Results on nuScenes test set. Besides the offline evaluation, we compare DynStaF with other SOTA single models on the nuScenes test server. No Test Time Augmentation (TTS) was used during the test phase. For a fair comparison, we compare with the results without TTS in Table 1. The methods are divided into two groups: (1) pillar-based methods which do not contain 3D convolutional operations; (2) voxel-based methods with 3D convolution blocks. As previous pillar-based methods usually deploy single-frame only as input, we also include a baseline for our re-implemented PointPillars with multi-frame as the input for a fair comparison. When using multi-frames, the vanilla PointPillar achieves 44.6% in mAP and 57.7% in NDS. With our DynStaF, PointPillars is improved by a large margin (5.9% in mAP and 3.9% in NDS), leading to the state-of-the-art performance for the pillar-based model: 50.5% for mAP and 61.6% for NDS. Moreover, notable improvement on individual object category can be observed. For example, on the traffic cone and barrier, DynStaF increases the mAP compared to the previous best results by 6.3% and 12.5%, respectively. Our DynStaF significantly strengthens pillar-based backbone, narrowing the performance gap compared to the voxel-based methods.

	mAP	**mAP** Gain	**NDS**	**NDS** Gain
PP* [11] (CVPR 19)	43.7	-	57.3	-
PP + DynStaF (ours)	49.4	**5.8**	61.4	**4.1**
CP* [26] (CVPR 21)	58.0	-	65.7	-
CP + VISTA [7] (CVPR 22)	57.6	1.2	65.6	0.8
CenterFormer [28] (ECCV 22)	55.4	0.2	65.2	0.8
CP + DynStaF (ours)	**58.9**	0.9	**67.1**	2.2

Table 1. Comparison with other SOTA methods on nuScenes validation set. * denotes our re-implementation backbone results. Result of mAP/NDS and its performance gain (compared to implemented backbones reported in previous works) are listed.

When comparing with other methods using 3D convolutional blocks, our DynStaF achieves the best performance in both mAP with 61.0% and NDS with 67.7%. When compared to the backbone method CenterPoint [26], DynStaF increases its performance in mAP by 3.0% and in NDS by 2.2%, which indicates its effectiveness. In particular, mAP of the construction vehicle or motorcycle is improved by a large margin. Overall, CenterPoint+DynStaF achieves the state-of-the-art performance on the nuScenes test set without bells and whistles.

4.3. Ablation study

In this section, we thoroughly analyze the effectiveness of each component, i.e., NCA, DSI and dual pathway architecture in our fusion strategy. All ablation studies are conducted on the NuScenes validation set and using the PointPillars [11] as the baseline model. The results are listed in Table 3. Using multi-frame point cloud as input, PointPillars without any feature fusion achieves 57.33% for NDS and 43.66% for mAP, respectively. If we use the naive feature fusion (denoted as "CNN-only") to replace the NCA module after each block, i.e., concatenating two features and adding CNN layers on top of it, the performance is improved to 59.74% NDS, which verifies that both feature branches have complementary information. With a more

	mAP	NDS	car	truck	bus	trailer	cons.	pedest.	motor.	bicycle	traff.	barrier
PointPillars [11]	30.9	45.3	68.4	23.0	28.2	23.4	4.1	59.7	27.4	1.1	30.8	38.9
WYSIWYG [9]	41.9	35.0	79.1	30.4	46.6	40.1	7.1	65.0	18.2	0.1	28.8	34.7
InfoFocus [20]	39.5	-	77.9	31.4	44.8	37.3	10.7	63.4	29.0	6.1	46.5	47.8
PMPNet [25]	-	45.4	79.7	33.6	47.1	43.0	18.1	76.5	40.7	12.3	58.8	48.4
PointPillars [11](Multi) *	44.6	57.7	80.3	44.7	55.5	47.6	11.4	69.7	27.7	5.4	54.4	49.7
PP + DynStaF (ours)	**50.5**	**61.6**	82.3	46.3	56.0	51.9	14.2	74.9	41.2	10.8	65.1	62.2
CGBS [29]	52.8	63.6	81.1	48.5	54.9	42.9	10.5	80.1	22.3	22.3	70.9	65.7
Pointformer [16]	53.6	-	82.3	48.1	55.6	43.4	8.6	81.8	55.0	22.7	72.2	66.0
CVCNet [4]	55.8	64.2	82.6	49.5	59.4	51.1	16.2	83.0	61.8	38.8	69.7	69.7
CenterPoint [26]	58.0	65.5	84.6	51.0	60.2	53.2	17.5	83.4	53.7	28.7	76.7	70.9
OHS [5]	59.3	66.0	83.1	50.9	56.4	53.3	23.0	81.3	63.5	36.6	73.0	71.6
CP + DynStaF (ours)	**61.0**	**67.7**	84.6	51.2	61.2	56.5	23.5	85.0	64.7	32.3	80.3	70.5

Table 2. Comparison with other SOTA (non-ensemble) LiDAR-based methods on nuScenes test set (without Test Time Augmentation). "cons.", "pedest.", "motor." and "traff." refer to construction vehicle, pedestrian, motorcycle and traffic cone, respectively. The first block is the methods not utilizing 3D convolutional networks, while the second block is. * denotes our re-implementation results.

sophisticated fusion module, our proposed NCA module, the NDS is improved further to 60.53% and mAP is increased by a large margin (4.27%) compared to the baseline. This indicates that using transformer-based cross attention mechanism is effective in the context of sparse point clouds. When the feature maps are dense, using DSI is more effective, as we see that NCA-only fusion is inferior to our final approach NCA + DSI.

We study the effectiveness of the dual-pathway architecture in the second block in Table 3. Instead of two feature streams, only one single pathway for feature fusion is deployed, i.e., the single-frame and multi-frame branch share weights of 2D CNN blocks highlighted by the color gray in Figure 2. The poor performance of this model (denoted as "Single") proves that it is impossible for a single backbone to deal with the single-frame and multi-frame features simultaneously. This also reveals that the multi-frame and single-frame contain different information. Our DynStaF ("Dual") arrives the best performance at 49.42% on mAP and 61.41% on NDS using all components, demonstrating the advantage of our proposed feature fusion strategy.

	mAP	NDS
Pointpillar* [11]	43.66	57.33
CNN-only fusion	47.49	59.74
NCA-only fusion	47.93	60.53
NCA + DSI (Single)	2.70	19.35
NCA + DSI (Dual)	**49.42**	**61.41**

Table 3. Ablation study results on nuScenes validation set. * denotes our re-implementation.

4.4. Analysis

Other cross-attention modules. Deformable DETR proposed in [30] learns to attend to a small set of keys around a reference point, which similarly discovers local attention as our NCA. The difference is that the sampling offsets (the position of keys) is learnable in Deformable attention module, while our NCA produces the "global" attention within a neighborhood. Deformable DETR has been proven to be efficient in the feature fusion based on LiDAR point clouds such as in [13, 28]. To discover the capability of the deformable attention in our use case, we replaced the NCA module in the pillar-based DynStaF with the Deformable DETR layer. However, we saw the performance degradation: NDS decreased to 60.04% and the mAP dropped to 47.01%. It indicates that the whole neighborhood is essential to build the cross attention between two branches.

Moreover, we also explored some other methods for the final interaction of features from two branches. For instance, we adapted the CBAM module [22] to learn the cross attention between two branches. When replacing DSI in the pillar-based DynStaF, NDS and mAP declined to 60.70% and 48.42%, respectively. These results show the advantages of our NCA and DSI in aggregating features and enhancing the interaction between two branches.

Efficiency of static branch. In the CenterPoint-based DynStaF, we aggregate the features in the 2D convolutional blocks and keep the channel dimensions in the two blocks the half as in the CenterPoint. We found this setting was not only efficient but also advantageous in the final performance. As we used an identical branch as the dynamic branch, i.e., a 3D convolutional backbone was used for single-frame input and the channel dimension was set to the same as in the original CenterPoint. We got 66.28% NDS

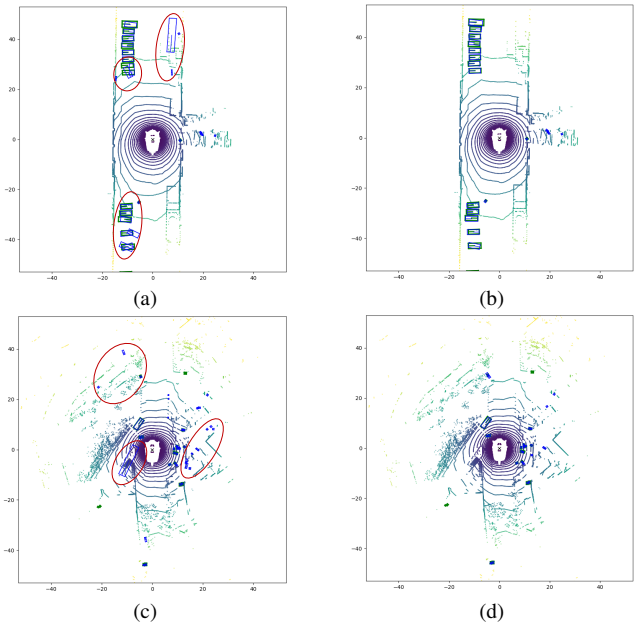


Figure 4. Visualization of object detection results on nuScenes validation set. Each row refers to one sample. (a) and (c): CenterPoint w/o DynStaF; (b) and (d): CenterPoint with DynStaF. Ground-truth bounding boxes are in green and prediction bounding boxes in blue.

and 58.41% mAP, which were inferior to the results of our DynStaF. The reason could be that the single-frame input was not sufficient to train a complex backbone. The current design of DynStaF provides lower computational cost and satisfactory performance in 3D detection at the same time.

4.5. Qualitative Results

We qualitatively show the advantage of DynStaF in the 3D object detection task. Figure 4 shows the prediction using CenterPoint as the backbone. In the first scene where there is a queue of vehicles on the left side. CenterPoint cannot detect the position of several vehicles (marked in the red circles) precisely, as shown in Figure 4a. DynStaF in Figure 4b localizes these vehicles correctly. Furthermore, DynStaF alleviates the false positives compared to the baseline. The second example is collected on a city street surrounded by many buildings with a group of pedestrian walking on the back right side of the car. As discussed in Figure 4a, multi-frame input is overwhelmed with point clouds in this case, making the detection difficult. For instance in Figure 4c, the point cloud of the walking pedestrians will be crowded such that the model predicts falsely (marked with the red circle on the right). With DynStaF, the single-frame can provide a clearer view of the each pedestrian as the point clouds are more sparse. These two examples show the advantage of our DynStaF in predicting location precisely and avoiding false positive detection.

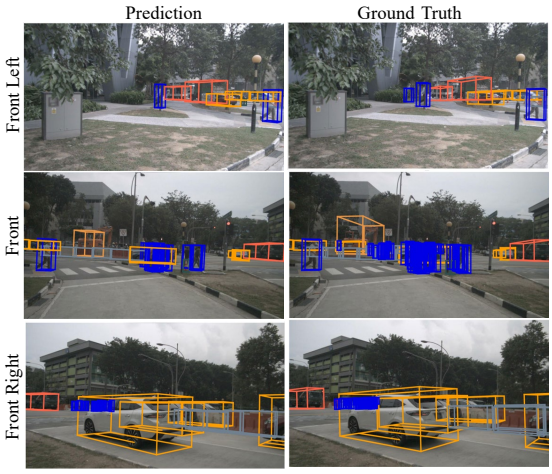


Figure 5. Visualization of object detection results on nuScenes validation set. Different rows represent different camera positions. The first column represents the prediction using CenterPoint+DynStaF; The second column is the ground-truth.

Figure 5 demonstrates the prediction in a camera view, which highlights concretely the ability of DynStaF in the context of dense point clouds. We show three challenging views where objects are closed to each other. In the front left and front right view, occlusion of vehicles can be observed, and our prediction is correct for most of the objects. In the front view, in which there exists more occlusion such as a group of walking pedestrians, DynStaF detects all objects but it cannot handle the occlusion position perfectly.

5. Conclusion

In this work, we propose a novel feature fusion framework named DynStaF to fuse the multi-frame and single-frame LiDAR point clouds for 3D object detection. Neighborhood Cross Attention module in DynStaF fuses features with a limited neighborhood instead of considering global attention, followed by Dynamic-Static Interaction module enhancing the feature interaction. Without loss of generality, our method can be utilized as a plug-and-play module in different pillar-based or voxel-based LiDAR point cloud detection algorithms. Quantitative results show that our DynStaF improves the strong backbone CenterPoint, outperforming other methods on the nuScenes dataset. Qualitatively, we demonstrate that our DynStaF can precisely localize objects thus avoid false positive predictions. Moreover, DynStaF enhanced the real-time pillar-based backbone significantly in the performance, highlighting its potential in practical usages. For future work, we aim to combine DynStaF with powerful frameworks taking LiDAR and camera images as input to further improve 3D object detection.

References

[1] Xuyang Bai, Zeyu Hu, Xinge Zhu, Qingqiu Huang, Yilun Chen, Hongbo Fu, and Chiew-Lan Tai. Transfusion: Robust lidar-camera fusion for 3d object detection with transformers. In *CVPR*, 2022. 2, 3, 5, 6

[2] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020. 1, 2, 5, 6

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 3

[4] Qi Chen, Lin Sun, Ernest Cheung, and Alan L Yuille. Every view counts: Cross-view consistency in 3d object detection with hybrid-cylindrical-spherical voxelization. 2020. 7

[5] Qi Chen, Lin Sun, Zhixin Wang, Kui Jia, and Alan Yuille. Object as hotspots: An anchor-free 3d object detection approach via firing of hotspots. In *ECCV*, 2020. 2, 7

[6] Qi Chen, Sourabh Vora, and Oscar Beijbom. Polarstream: Streaming object detection and segmentation with polar pillars. 2021. 1, 2

[7] Shengheng Deng, Zhihao Liang, Lin Sun, and Kui Jia. Vista: Boosting 3d object detection via dual cross-view spatial attention. In *CVPR*, 2022. 1, 2, 3, 5, 6

[8] Ali Hassani, Steven Walton, Jiachen Li, Shen Li, and Humphrey Shi. Neighborhood attention transformer. *arXiv preprint arXiv:2204.07143*, 2022. 3, 4

[9] Peiyun Hu, Jason Ziglar, David Held, and Deva Ramanan. What you see is what you get: Exploiting visibility for 3d object detection. In *CVPR*, 2020. 7

[10] Dihe Huang, Ying Chen, Yikang Ding, Jinli Liao, Jianlin Liu, Kai Wu, Qiang Nie, Yong Liu, and Chengjie Wang. Rethinking dimensionality reduction in grid-based 3d object detection. *arXiv preprint arXiv:2209.09464*, 2022. 2, 3

[11] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019. 1, 2, 5, 6, 7

[12] Yanwei Li, Yilun Chen, Xiaojuan Qi, Zeming Li, Jian Sun, and Jiaya Jia. Unifying voxel-based representation with transformer for 3d object detection. 2022. 1, 2, 3

[13] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Qiao Yu, and Jifeng Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. *arXiv preprint arXiv:2203.17270*, 2022. 3, 7

[14] Zhijian Liu, Haotian Tang, Alexander Amini, Xinyu Yang, Huizi Mao, Daniela Rus, and Song Han. Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation. *arXiv preprint arXiv:2205.13542*, 2022. 2

[15] Jongyoun Noh, Sanghoon Lee, and Bumsub Ham. Hvpr: Hybrid voxel-point representation for single-stage 3d object detection. In *CVPR*, 2021. 3

[16] Xuran Pan, Zhuofan Xia, Shiji Song, Li Erran Li, and Gao Huang. 3d object detection with pointformer. In *CVPR*, 2021. 7

[17] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. 2017. 3

[18] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, 2020. 2

[19] OpenPCDet Development Team. Openpcdet: An open-source toolbox for 3d object detection from point clouds. https://github.com/open-mmlab/OpenPCDet, 2020. 6

[20] Jun Wang, Shiyi Lan, Mingfei Gao, and Larry S Davis. Infofocus: 3d object detection for autonomous driving with dynamic information modeling. In *ECCV*, 2020. 1, 2, 7

[21] Yue Wang, Alireza Fathi, Abhijit Kundu, David A Ross, Caroline Pantofaru, Tom Funkhouser, and Justin Solomon. Pillar-based object detection for autonomous driving. In *ECCV*, 2020. 1, 2

[22] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *ECCV*, 2018. 7

[23] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 2018. 1, 2

[24] Dongqiangzi Ye, Zixiang Zhou, Weijia Chen, Yufei Xie, Yu Wang, Panqu Wang, and Hassan Foroosh. Lidarmultinet: Towards a unified multi-task network for lidar perception. *arXiv preprint arXiv:2209.09385*, 2022. 1

[25] Junbo Yin, Jianbing Shen, Chenye Guan, Dingfu Zhou, and Ruigang Yang. Lidar-based online 3d video object detection with graph-based message passing and spatiotemporal transformer attention. In *CVPR*, 2020. 7

[26] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *CVPR*, 2021. 2, 3, 5, 6, 7

[27] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, 2018. 1, 2

[28] Zixiang Zhou, Xiangchen Zhao, Yu Wang, Panqu Wang, and Hassan Foroosh. Centerformer: Center-based transformer for 3d object detection. In *ECCV*, 2022. 3, 6, 7

[29] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3d object detection. *arXiv preprint arXiv:1908.09492*, 2019. 2, 6, 7

[30] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *ICLR*, 2020. 3, 7